Attacks and Countermeasures in Fingerprint Based Biometric Cryptosystems

Benjamin Tams *

April 30, 2013

Abstract

We investigate implementations of biometric cryptosystems protecting fingerprint templates (which are mostly based on the *fuzzy vault scheme* by Juels and Sudan in 2002) with respect to the security they provide. We show that attacks taking advantage of the system's false acceptance rate, i.e. *false-accept attacks*, pose a very serious risk — even if *brute-force attacks* are impractical to perform. Our observations lead to the clear conclusion that currently a single fingerprint is not sufficient to provide a secure biometric cryptosystem. But there remain other problems that can not be resolved by merely switching to multi-finger: Kholmatov and Yanikoglu in 2007 demonstrated that it is possible to break two matching vault records at quite a high rate via the *correlation attack*.

We propose an implementation of a minutiae fuzzy vault that is inherently resistant against *cross-matching* and the correlation attack. Surprisingly, achieving cross-matching resistance is not at the cost of authentication performance. In particular, we propose to use a randomized decoding procedure and find that it is possible to achieve a GAR = 91% at which no false accepts are observed on a database generally used. Our ideas can be adopted into an implementation of a multibiometric cryptosystem. All experiments described in this paper can fully be reproduced using software available for download.[1]

Keywords

fingerprint, fuzzy vault, cryptanalysis, cross-matching, correlation attack

*Benjamin Tams is with the Institute for Mathematical Stochastics, University of Goettingen, Goldschmidtstr. 7, 37077, Goettingen, Germany. Phone: +49-(0)551-3913515. Email: btams@math.uni-goettingen.de

[1]Source code of the programs can be downloaded along with source code of a C++ library `thimble` from `http://www.stochastik.math.uni-goettingen.de/biometrics`.

1 Introduction

In a traditional password-based authentication scheme, user names along with their respective passwords are stored on a server-side database. In such a scenario, we usually cannot prevent the fact that some persons will have access to the content of the database. Such persons (for example, system administrators) are thus able to read the password information related to enrolled users. To prevent them from using password information to impersonate authorized users, a non-invertible transformation[2] (e.g., a *one-way hash function*) of each password is stored rather than the unprotected password. During authentication, the user sends his password to the service provider, which computes the password's non-invertible transformation. Next, the service provider compares the transformation that is stored in the database with the just-transformed password. If both agree, the authentication attempt is accepted; otherwise it is rejected. So a would-be thief cannot find the passwords in the database; he must either steal them from users or guess them.

However, a password can be forgotten or, if written down, can be stolen. To prevent these risks, many individuals attempt to create easily memorable passwords. Unfortunately, this often results in the individual choosing a weak password, typically constructed using personal information (e.g., a birthday or the name of a significant other), which increases the risk of others' guessing the password — and therefore, of theft.

A popular alternative to password is to base authentication on biometrics such as fingerprints. Authentication protocols that incorporate biometric templates do not have the disadvantage that they can be forgotten or lost: Barring injuries, fingers are always with us; moreover, fingerprint features remain reasonably invariant over time.

If a biometric authentication scheme is in place the incorporated templates have especially to be stored protected. The situation is rather serious, because biometric templates may correspond to human beings with nearly unique precision compared to mere passwords. In addition, ineffective protection of biometric templates has consequences beyond breach of privacy, as compared to protecting passwords: For example, if a password is corrupted (e.g., discovered by others) it can be replaced easily compared to replacing a biometric template.

While the requirements for so-called *biometric template protection schemes* are similar to those used for protecting passwords, they are more difficult to achieve: With high confidence it must be efficiently verifiable whether a provided biometric template matches the template that is encrypted by the stored data; furthermore, it must be computationally infeasible to de-

[2]Non-invertible transformation of a password means that it is easy to transform the password, while on the contrary the derivation of a password from a given transformation is computationally hard.

rive the unencrypted biometric template from the stored data. There is one great difference between password and biometric authentication schemes: Contrary to passwords different measurements of the same biometric source will differ, while also having some reasonable similarity. These differences between two biometric templates of the same individual can be usefully conceptualized as deviations or errors. In this vein, there have been proposals for biometric template protections schemes that couple techniques from traditional cryptography with techniques from the discipline of *error-correcting codes*.

1.1 The Fuzzy Vault Scheme

In 2002, Juels and Sudan proposed the *fuzzy vault scheme* [1] which is a construction for protecting noisy data. While the *fuzzy commitment scheme* [2], which was proposed by Juels and Wattenberg in 1999, requires the data to be presented as a fixed-length feature vector, the fuzzy vault scheme allows the length of the data to vary and the data to be unordered. These properties enable the fuzzy vault scheme to protect fingerprint templates such as fingerprint minutiae. It works as follows.

Enrollment

Given a fingerprint template containing t fingerprint features, e.g., minutiae, its elements are encoded as elements x in a fixed finite field $\mathbf{F}$. One chooses a secret message polynomial $f \in \mathbf{F}[X]$ in the indeterminate X of degree smaller than k and evaluates $f(x)$ at the encoded element x. The genuine pairs $(x, f(x))$ are dispersed among a large set of chaff points that do not lie on the graph of f, such that a vault of size n is built.

Authentication

Using a second genuine template one aims at distinguishing the genuine points from the chaff points. Given the points are mainly genuine, one can tolerate errors within certain limits determined by error-correcting codes [3–6].

Security

From the difficulty of the problem of distinguishing genuine from chaff (without the help of a second genuine template) the fuzzy vault scheme draws its security. This problem can be reduced from the *polynomial reconstruction problem* which is believed to be hard in general if $t \ll \sqrt{(k-1) \cdot n}$ [7–11].

1.2 The Fuzzy Fingerprint Vault

There are several biometric traits which can be used in biometric authentication systems (see [12] for an overview). Every biometric discipline is associated with very individual challenges which have to be solved before incorporating them into a biometric template protection scheme. One biometric trait that can be extracted from humans are his *fingerprints* [13]. This papers focuses on fingerprints and examines their adequacy of being protected by the fuzzy vault scheme. Even though there are other biometric template protection schemes [14] and implementations for fingerprints [15] the fuzzy vault scheme is the most dominant and promising scheme for which implementations to protect fingerprints have been proposed.

Implementations

Several fuzzy vault variants for protecting fingerprint minutiae templates can be found in the literature [16–23].

To increase vault practicability as well as security, Nagar et al. (2008, 2010) [22, 23] proposed to fuse a fingerprint's minutiae template with information about its ridge orientation and frequency by means of *minutiae descriptors* [24]. In distinguishing genuine from chaff minutiae an attacker has in addition to guess the respective minutiae descriptors. This adds some security to the base vault implementation.

All the implementations above require an *alignment step* where the query minutiae templates are aligned to the vault. This is very challenging since the enrolled templates are protected. Currently, the alignment is realized by techniques using auxiliary alignment data, e.g., see [19, 20]. The use of auxiliary alignment data, however, may cause security issues resulting in the leakage of information from the protected fingers.

Another interesting approach is to use *alignment-free features*, i.e. features that do not dependent of the finger's rotation or displacement. Li et al. (2010) [25] proposed to fuse *minutiae local structures* [26] with *minutiae descriptors* [24] and to protect them by the fuzzy vault scheme. The recognition performance that the authors report is promising and the error-prone step of aligning the query fingerprint to the vault is circumvented. Furthermore, the problem of information leakage by auxiliary alignment data does not exist anymore.

1.3 Content and Contribution of the Paper

After we described the functioning of a minutiae fuzzy vault in more detail (Section 2) we investigate the security of implementations from the literature in different attack scenarios (Section 3). Reproducing the work of Mihăilescu et al. (2009) [27] we show that brute-force attacks can be very practical to perform against most implementations (see Section 3.1).

But even if brute-force attacks are infeasible to perform, there remains the possibility of the attacker to run an attack that takes it advantage out of the system's *false acceptance rate*, i.e. *false-accept attack*; also see Section 26.6.1.1 in [28]. We show that false-accept attacks are even much easier to perform than brute-force attacks (see Section 3.3). Note that the false-accept attack is not restricted to the fuzzy vault scheme but it can be applied with virtually no modifications to every authentication scheme. Its attack success rate only depends on the system's false acceptance rate and the average time needed to run an impostor recognition attempt. Therefore, our observations clearly advocate that biometric cryptosystems merely based on a single finger cannot provide effective security. Rather multi-finger cryptosystems [29] (or even *multibiometric cryptosystems* [30]) should be developed.

For the fuzzy vault scheme there remains a problem that can not be solved merely by switching to multibiometrics. Given two matching instances of a minutiae fuzzy vault to an adversary he can correlate them; genuine minutiae tend to agree well in comparison to chaff minutiae, which are likely to be in disagreement. Thus, an intruder may reliably determine whether two vault records match, i.e. *cross-matching*. Even worse, via correlation the adversary can try to distinguish genuine minutiae from chaff minutiae. If in this way a set of vault minutiae can be extracted that contains a reasonable proportion of genuine minutiae, then the vault can efficiently be broken. Consequently, this attack is called *correlation attack*. Scheirer and Boult (2007) were the first who have drawn the attention to the risk of attacking fuzzy vault via record multiplicity [31]. Then Kholmatov and Yanikoglu (2008) have demonstrated the practicability of the correlation attack [32]. Therefore, in Section 4, we propose an implementation of a minutiae fuzzy vault that is inherently resistant against cross-matching and the correlation attack. Fortunately, cross-matching resistance can be achieved without decreasing the verification performance as we found in a test on a fingerprint database publicly available (see Section 4.4). This is mainly due to a randomized decoding procedure that we propose.

A final discussion, conclusion, and an outlook are given in Section 5.

All experiments described in this paper can fully be reproduced using software that we made available for download.[1]

2 Minutiae Fuzzy Vault Implementation

Assume tha we are given a minutiae template $\{(a, b, \theta)\}$ where (a, b) and θ denote its position and angle, respectively. Using the fuzzy vault scheme we may protect the template as follows.

2.1 Enrollment

We describe the vault construction analogous to Nandakumar et al. (2007) [20] with some minor modifications.

As in [20], only well-separated minutiae are selected. Furthermore, only the $t \leq t_{\max}$ minutiae of best quality that are well-separated are selected for vault construction. If it is not possible to select at least a certain number of $t_{\min}$ minutiae, the enrollment is aborted and a *failure to capture* is reported. Otherwise, the construction continues as follows. To hide the selected *genuine minutiae* T_{gen}, a set of *chaff minutiae* T_{chaff} is generated at random fulfilling the following side conditions: First, each chaff minutia has the property that it is well-separated from all other vault minutiae — genuine and chaff; second, a chaff minutia's position lays within the corresponding fingerprint image's region; third, the number of chaff minutiae is such that the vault minutiae reach a predefined size $n \geq t$, i.e. $n - t$ chaff minutiae are generated. The union of genuine and chaff minutiae is referred to as the *vault minutiae* $T_{\text{vault}} = T_{\text{gen}} \cup T_{\text{chaff}}$.

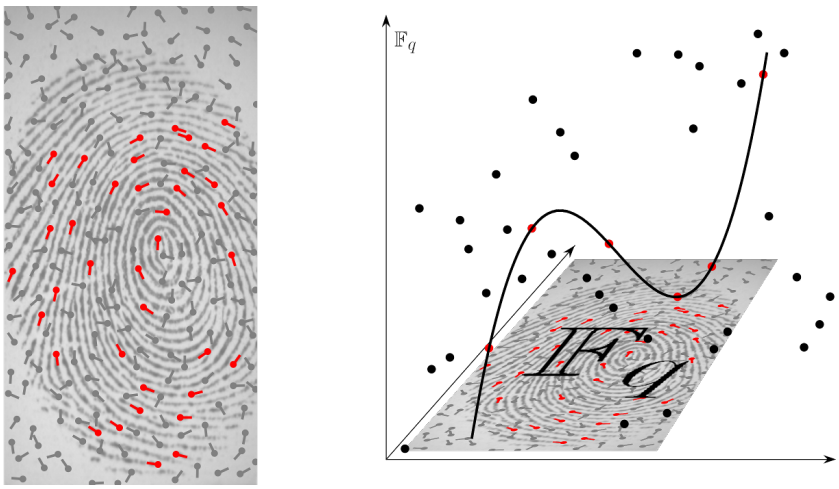
After the vault minutiae have been established, a secret is encoded as a polynomial f of degree $< k$ having coefficients in a fixed finite field $\mathbf{F} = \mathbb{F}_q$ of size $q \geq n$. Now, list the vault minutiae as $(a_0, b_0, \theta_0), \ldots, (a_{n-1}, b_{n-1}, \theta_{n-1})$ by some convention, e.g., by sorting them in lexicographical order. By $x_0, \ldots, x_{n-1} \in \mathbf{F}$ denote n distinct elements of the finite field. In this way, each list index $i = 0, \ldots, n-1$ uniquely encodes an element in $\mathbf{F}$. Now we build the *genuine set* as $\mathbf{G} = \{(x_i, f(x_i)) \mid (a_i, b_i, \theta_i) \in T_{\text{gen}}\}$. Analogously, the *chaff set* is defined as $\mathbf{C} = \{(x_j, y_j) \mid (a_j, b_j, \theta_j) \in T_{\text{chaff}}\}$ where the y_j s are chosen uniformly at random such that $y_j \neq f(x_j)$. The union $\mathbf{V} = \mathbf{G} \cup \mathbf{C}$ builds the *vault points*.

The protected template is published as the triple $(\mathbf{V}, T_{\text{vault}}, h(f))$ where $h(f)$ denotes a cryptographic hash value of f (e.g., SHA-1) to allow safe recovery of f at genuine authentication.

Note, that there is a one-to-one correspondence between the vault $\mathbf{V}$ and the vault minutiae T_{vault}. Thus, given a genuine minutia we also know its corresponding vault point and vice versa. In this respect, our construction is different from the construction of Nandakumar et al. (2007) who encode the minutiae information on the x-coordinate of its corresponding vault point. Another difference of our construction is that we use a SHA-1 hash value instead of constituting the secret polynomial with redundancy bits.

2.2 Authentication

On authentication, a query template of the (alleged) genuine user is provided. As on enrollment, only well-separated minutiae of good quality are selected (again, at most $t_{\max}$). For simplicity, we assume that the query minutiae are correctly aligned to the vault. We extract those vault minu-



(a) (b)

Figure 1: (a) Genuine (red) and chaff minutiae (gray); (b) each minutia is encoded on a vault point's abscissa where its ordinate binds the minutia to the secret polynomial

tiae that are well approximated by aligned query minutiae. In this way, we establish the *unlocking set* $\mathbf{U}$ which consists of those vault points that correspond to the just extracted minutiae. Let t be the size of the unlocking set $\mathbf{U}$. There are $\binom{t}{k}$ combinations of selecting k different unlocking points. For each combination, the interpolation polynomial $f^* \in \mathbf{F}[X]$ is computed and it is checked whether its hash value agrees with the hash value of the correct polynomial, i.e. if $h(f^*) = h(f)$. If true then $f^* = f$ with overwhelming reliability and f^* is output as the correct polynomial which corresponds to a successful authentication. Otherwise, if all $h(f^*) \neq h(f)$ the authentication attempt is rejected.

2.3 Alignment

There have been proposals to ease aligning the query minutiae to the vault such that matching vault minutiae agree with their respective query minutiae (see [17, 19, 20, 33, 34]). All of these proposals leak information about the corresponding fingerprint, e.g., about some of its minutiae or its orientation field. Moreover, it is not clear to what extent auxiliary alignment data can help an adversary to find matching vault correspondences via cross-matching (see Section 3.5).

Ideally, fingerprints can be pre-aligned such that matching query minutiae already agree with genuine vault minutiae. However, pre-alignment is currently not very robust. But increasing robustness of fingerprint pre-

alignment would automatically increase the practicability of a minutiae fuzzy vault implementation without decreasing its overall security. Therefore, although challenging, it seems to be worth to search for more robust pre-alignment procedures. Alternatively, suitable alignment-free features can be used for constructing the vault (see [25]).

Due to open questions related to vault alignment, if we investigate vault performances, we assume a well-solved alignment framework for genuine authentication. Consequently, if an authentication of a genuine user is simulated, the alignment is obtained by aligning the query minutiae template to the enrolled template in clear. On an impostor authentication, we do not make any attempts in aligning the query template to the vault.

2.4 Evaluation Database and Protocol

Throughout, we used the FVC 2002 DB2 database[3] for our performance evaluations as it is the common database used to evaluate fingerprint fuzzy vault implementations.

We strictly follow the FVC protocol [35] even though in the literature the implementations are evaluated following a protocol in where the number of observed impostor recognition attempts is artificially increased [18–20,22,23, 25]. But this would not correspond to statistically independent observations.

As already described in Section 2.3, on an genuine authentication attempt we assert that the query finger is correctly aligned by aligning both fingers in clear; for an impostor recognition attempt, we do not make any attempts for alignment.

Genuine acceptance rates and false acceptance rates will be denoted by GAR and FAR, respectively. Furthermore, throughout the literature the genuine acceptance rate incorporating the first two impression of each fingers only are reported. This corresponds to the scenario in where the fingerprints are of good quality which positively affects the genuine acceptance rates. Therefore, to allow for comparing our genuine acceptance rates with other implementations, we will also keep track of the genuine acceptance rate w.r.t. the subset of the database. The corresponding genuine acceptances rates are indicated by sub-GAR.

The minutiae templates that we used have been obtained using a commercial extractor.[4]

2.5 Performance Evaluation

We evaluated the vault performances for different k on the FVC 2002 DB2 database following the FVC protocol (see [35]) using parameters adopted from Nandakumar et al. (2007). They propose to hide at most $t_{\max} = 24$

[3] The database consists of 8 impressions each acquired from a total of 100 fingers.

[4] Verifinger SDK 5.0 [36]

Table 1: Performance Evaluation of our minutiae fuzzy vault re-implementation using parameters adopted from Nandakumar et al. (2007) [20]

polynomial degree $< k$	genuine acceptance rate GAR (sub-GAR)	false acceptance rate FAR	avg. genuine decoding time GDT	avg. impostor decoding time IDT
$= 7$	$\approx 86.54\%$ ($\approx 96\%$)	$\approx 3.87\%$	≈ 0.05 *sec*	≈ 0.08 *sec*
$= 8$	$\approx 80.76\%$ ($\approx 92\%$)	$\approx 1.63\%$	≈ 0.121 *sec*	≈ 0.140 *sec*
$= 9$	$\approx 74.61\%$ ($\approx 92\%$)	$\approx 0.56\%$	≈ 0.226 *sec*	≈ 0.198 *sec*
$= 10$	$\approx 67.52\%$ ($\approx 92\%$)	$\approx 0.16\%$	≈ 0.351 *sec*	≈ 0.240 *sec*
$= 11$	$\approx 58.93\%$ ($\approx 91\%$)	$\approx 0.10\%$	≈ 0.5 *sec*	≈ 0.248 *sec*
$= 12$	$\approx 51.07\%$ ($\approx 87\%$)	$= 0\%$	≈ 0.546 *sec*	≈ 0.193 *sec*

and at least $t_{\min} = 18$ well-separated[5] genuine minutiae in a vault of size $n = 224$.

Example 1. *For example, if* $k = 9$ *then the genuine acceptance rate was determined as* GAR $\approx 74.61\%$ *and the false acceptance rate as* FAR $\approx 0.56\%$. *The total number of genuine authentication attempts and impostor authentication attempts was* 2749 *and* 4856, *respectively.* FTCR $\approx 3.88\%$ *of the enrollments were aborted, because it was not possible to select at least* $t_{\min} = 18$ *well-separated minutiae.*

The genuine acceptance rate measured on the finger's respective first two impressions was found to be sub-GAR $\approx 92\%$ *at a failure to capture rate of* sub-FTCR $= 1\%$.

Note that our rates differ from those reported in [20]. This is due to the use of a different authentication scheme than [20] as well of the fact that we decoupled the alignment from the vault.

We will use the re-implementation to demonstrate the effectiveness of the false-accept attack in Section 3.3. Note, even in the case that we would use auxiliary automatic alignment data to ease alignment, an attacker does not have to account for it; our false acceptance rates reflect the success rate of such a corresponding attack.

[5] Two minutiae (a, b, θ) and (a', b', θ') are said to be well-separated if $\|(a, b) - (a', b')\|_2 + 0.2 \cdot \max(|\theta - \theta'|, |360^\circ - \theta + \theta'|) > 25$.

2.6 Fuzzy Vault with Minutiae Descriptors

To improve the practicability as well as the security of the construction in [20], in addition to mere minutiae, Nagar et al. (2008, 2010) [22, 23] proposed to incorporate minutiae descriptors in constructing the vault.

A minutia's descriptors consists of the ridge orientation (relative to the orientation of the minutia) and ridge frequency of points arranged around the minutia (see Fig. 2). The authors showed how a minutia descriptor can be quantized as an m-bit vector $w \in \{0,1\}^m$. Furthermore, the corresponding vault points $(x, y) \in \mathbf{F} \times \mathbf{F}$ ordinate value y is encoded as a codeword $c(y)$ of a binary error-correcting code of length m which is capable in correcting ν (say) errors. The *fuzzy commitment* [2] of $c(y)$ using the witness w is computed next, i.e. $c(y) + w$.[6] Rather publishing the vault point (x, y) the tuple $(x, c(y) + w)$ is published instead. For chaff points the ordinate values are protected using random descriptor binarizations from a pool of chaff descriptors.

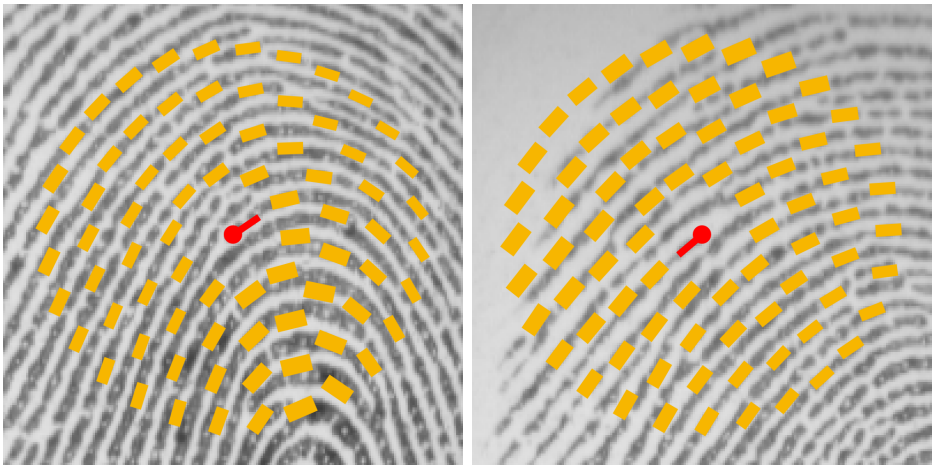


Figure 2: *Minutiae descriptors* — thickness and orientation of yellow lines correspond to ridge frequency and orientation descriptor, respectively; the orientation fields and frequency images to visualize were estimated using the methods in [37] and [38], respectively.

On authentication, the unlocking points $(x, c(y) + w)$ are extracted as in the basic vault. Using the minutiae descriptor $w' \in \{0,1\}^m$ of the corresponding query minutia, the difference $c(y) + w - w'$ is computed.[7] If w' is sufficiently similar to w, i.e. if they differ in at most ν positions, the difference $c(y) + w - w'$ can be corrected to $c(y)$ which encodes the correct y. Therefore, in addition to sufficiently many genuine vault points among the

[6] Addition is performed bitwise modulo 2 which is equivalent to a bitwise xor operation.

[7] Note that addition modulo 2 is the same as subtraction modulo 2, i.e. $c(y) + w - w' = c(y) + w + w'$.

unlocking set it is required that their correct ordinate values can be recovered. This will be the case, if a minutia descriptor with sufficient similarity to the genuine descriptor can be found. Thus, for the vault to successfully unlock there is required more agreeing information of the query template to the enrolled template and thus the basic vault's security is improved.

Next, we investigate the security of different fingerprint fuzzy vault implementations from the literature.

3 Attacks

3.1 Brute-Force Attack

While attack scenarios involving *brute-force attacks* are analyzed throughout the literature they frequently lack of emphasizing how practical these naive attacks can be. In this section, for parameters adopted from implementations found in the literature we determine the expected number of computer time that is expected to be required for a successful brute-force attack. Therefore, we briefly reproduce the work of Mihăilescu et al. (2009) [27] to emphasize the practicability of brute-force attacks against current implementations of the fuzzy fingerprint vault; for a smart polynomial reconstruction approach we refer to Choi et al. (2011) [39]. Afterwards we modify the attack for the implementation of Nagar et al. (2008, 2010) [22, 23].

Assume that an intruder has intercepted a vault of size n in which t genuine vault points are contained laying on the graph of a common polynomial of degree $< k$. Furthermore, we assume that a cryptographic hash value $h(f)$ of the correct polynomial is publicly available to the adversary. To find the correct polynomial, the intruder 1) may guess k random vault points, 2) determine its interpolation polynomial f^*, and 3) check whether $h(f^*) = h(f)$: If true, the attacker has found the correct polynomial with overwhelming reliability; otherwise, he repeats the attack until a polynomial f^* with $h(f^*) = h(f)$ is found.

The probability that a random choice of k vault points yields the correct polynomial is $\mathbf{bf}(n, t, k)^{-1}$ where

$$\mathbf{bf}(n, t, k) = \binom{n}{k} \cdot \binom{t}{k}^{-1}. \tag{1}$$

Thus, after

$$\log(0.5)/\log(1 - \mathbf{bf}(n, t, k)^{-1}) \tag{2}$$

iterations the adversary can expect to find the correct polynomial.

Example 2. *For example, if* $(n, t, k) = (224, 24, 9)$ *(which are parameters as proposed by Uludag and Jain 2006 [19]) the adversary can expect to successfully break the vault after* $\approx 2^{31}$ *iterations using Formula (2). For* $\mathbf{F} = \mathbb{F}_{2^{16}}$

Table 2: Expected computational timings for running a successful brute-force attack on a 3.2 Ghz desktop computer with four processor cores against different vault parameters found in the literature

parameters as in	(n, t, k)	security	iterations per second per core	**expected time for success**
[18]	$(218, 18, 9)$	$\approx 2^{36}$	$151,316.5$	< 17 *hours*
[19]	$(224, 24, 9)$	$\approx 2^{31}$	$148,634.1$	< 50 *min*
[20]	$(224, 24, 8)$	$\approx 2^{27}$	$183,188.8$	< 3 *min*
"	$(224, 24, 9)$	$\approx 2^{31}$	$148,634.1$	< 50 *min*
"	$(224, 24, 11)$	$\approx 2^{39}$	$109,066.9$	< 11 *days*
[34]	$(440, 40, 13)$	$\approx 2^{48}$	$82,056.84$	< 18 *years*
"	$(440, 40, 14)$	$\approx 2^{52}$	$69,227.56$	< 325 *years*

we experimentally determined that it is possible to perform $148,634.1$ *iterations in one second of the above brute-force attack. Thus, if four processors/cores are used by the attacker he can expect to be successful after* ≈ 49 *min.*

We empirically determined expected times for a successful brute-force attack for parameters from different implementations from the literature. The results are listed in Table 2. We find that brute-force attacks can become practical to perform easily — even on a standard desktop computer. Related work can be found in [27, 39].

3.2 Attack against Fuzzy Vault with Minutiae Descriptors

To break instances of the implementation of Nagar et al. (2008,2010) [22,23] the attacker must act differently in choosing a candidate polynomial f^* because the vault points ordinate values are protected. Therefore, we assume that the adversary has access to a large pool of minutiae descriptors.

3.2.1 Decoupling the Vault from Protected Ordinate Values

For each protected vault point $(x, c(y) + w)$ (chaff and genuine) the attacker iterates through the descriptor pool. For each descriptor $w' \in \{0, 1\}^m$ the

difference $c(y) + w - w'$ is computed and then an attempt is made to decode $c(y) + w - w'$ to its nearest codeword $c(y')$. There are three possible cases:

i) The attacker can correct to the right $c(y)$ and thus obtains the correct ordinate value y;

ii) he obtains another codeword $c(y') \neq c(y)$ and thus an incorrect ordinate value $y' \neq y$;

iii) the difference cannot be corrected to any codeword.

While iterating through the descriptor pool, the attacker establishes a set of candidate ordinate values. For simplicity, we assume that the correct ordinate value can be found in the candidate set for each vault point. By $\{y'\}$ denote such a candidate set. For the attacker to select a candidate polynomial, he may randomly choose k distinct vault points and, in addition, for each vault point a random candidate ordinate value.

To estimate the probability that such a candidate polynomial yields the correct polynomial, we first estimate the expectation of the size of the candidate set for random vault points $(x, c(y) + w)$.

An important tool to achieve this is the *sphere packing density* of the underlying binary error-correcting code, which can be defined to be the probability that a random m-bit word can be corrected to a valid codeword. Therefore, by ℓ denote the number of codewords. Then its sphere packing density is

$$\rho = 2^{\ell - m} \sum_{j=0}^{\nu} \binom{m}{j} \tag{3}$$

where ν denotes the code's error-correcting capability. We argue with [23] that the difficulty in guessing a random minutiae descriptor is $R \approx 4.27$. Therefore, we estimate the expectation of the number of of candidate ordinate values for each vault point as $S = 1 + (R - 1) \cdot \rho$. Thus, we estimate the brute-force security as

$$S^k \cdot \mathbf{bf}(n, t, k). \tag{4}$$

3.2.2 Evaluation of the Attack

For the implementation of [23] in where the ordinate values are protected via a $(511, 19)$-BCH code, which can correct $\nu = 119$ errors, the corresponding sphere packing density is $\rho \approx 1.3 \cdot 10^{-29}$. Thus, we expect the number of a vault point's candidate ordinate values to be $S = 1 + (R - 1) \cdot \rho \approx 1 + 4.25 \cdot 10^{-27}$. Consequently, the estimated brute-force security for vault parameters $(n, t, k) = (224, 24, 9)$ is $S^k \cdot \mathbf{bf}(n, t, k) \approx 2.54 \cdot 10^9$ which corresponds to 31 bits. In comparison, the brute-force security for the base implementation without protected ordinate values is $\mathbf{bf}(n, t, k) \approx 2.54 \cdot 10^9$ which is almost

the same. Thus, there is virtually no improvement in protecting the vault points ordinate values if the code's sphere packing density is too small. As a countermeasure, the use of two other BCH codes of higher sphere packing density have been investigated in [23] with a measurable improvement in the brute-force security. The corresponding evaluations can be found in Table 3.

Table 3: Brute-force securities of the implementation of Nagar et al. (2010) for different choices of BCH codes and for different polynomial degrees. The genuine acceptance rates have been extracted from Figure 7 in [23] in where the false acceptance rates have been indicated as to be very close to 0.

	BCH(511, 19)		BCH(31, 6)		BCH(15, 5)	
polynomial degree	brute-force security	sub-GAR	brute-force security	sub-GAR	brute-force security	sub-GAR
$k = 7$	$\approx 2^{24}$	95%	$\approx 2^{27}$	94%	$\approx 2^{34}$	93%
$k = 8$	$\approx 2^{27}$	94%	$\approx 2^{31}$	93%	$\approx 2^{40}$	93%
$k = 9$	$\approx 2^{31}$	93%	$\approx 2^{35}$	93%	$\approx 2^{45}$	91%
$k = 10$	$\approx 2^{35}$	89%	$\approx 2^{39}$	87%	$\approx 2^{50}$	85%
$k = 11$	$\approx 2^{39}$	84%	$\approx 2^{44}$	81%	$\approx 2^{56}$	76%
$k = 12$	$\approx 2^{43}$	78%	$\approx 2^{48}$	77%	$\approx 2^{61}$	73%

3.3 False-Accept Attack

Brute-force attacks can definitely be improved. For example, one may use statistics of fingerprints to accelerate brute-force attacks. Assuming that the statistics of fingerprints is best reproduced by real fingers, making heuristic considerations one may conclude that an attack that takes advantage out of the system's false-acceptance rate FAR yields the system's overall security. In any case, such an attack yields an upper bound of the system's overall security and is a hint for the existence of a similar efficient statistical attack.

In the scenario of a false-accept attack we assume that an adversary who has intercepted a vault also has access to a sufficiently large database containing fingerprint templates.

Then the adversary may try to recover the protected template from the vault off-line by simulating authentication attempts using the templates in

the database as the queries. For a random query template, with probability FAR the vault will unlock and reveal the protected key and template. Thus, the adversary can expect to successfully break the vault after he has simulated $\log(0.5)/\log(1-\mathrm{FAR})$ authentication attempts. If the average impostor decoding time IDT is known then the computational cost for a successful false-accept attack can be estimated as

$$\log(0.5)/\log(1-\mathrm{FAR}) \cdot \mathrm{IDT}\,. \tag{5}$$

Example 3. *For example, assume that an adversary has intercepted a minutiae fuzzy vault as in Section 2 with* $k = 9$*. With Table 1 we may assume that* $\mathrm{FAR} \approx 0.56\%$ *and* $\mathrm{IDT} \approx 0.198$ *sec. Thus, the adversary can expect to successfully break the vault after only* ≈ 24.6 *sec. If four processors/cores are used in parallel the time furthermore reduces to approximately* 6.15 *sec. In comparison to the brute-force attack, which takes* ≈ 49 *min on the same computer, the false-accept attack turns out to be the better choice for the adversary and thus poses the more serious risk.*

3.3.1 Confidence of the False Acceptance Rate

In the above example we assumed that the false acceptance rate was $\mathrm{FAR} \approx 0.56\%$. This is because we observed 27 false accepts among $4,856$ simulated impostor authentication attempts and thus $\mathrm{FAR} = 27/4,856 \approx 0.56\%$. But actually, the observation of a false accept is the result of a random sample.

Assume that we observed s false accepts among N impostor recognition attempts. Let $\mathrm{FAR}^* = s/N$ be the *point estimation* for the false acceptance rate. We can only be absolutely certain that $\mathrm{FAR} \in (0\%, 100\%)$ but, roughly speaking, it is not very likely that the true false acceptance rate differs from FAR^* too much. To estimate the confidence of FAR^*, a useful concept is the one of *confidence intervals*.

Definition 1 (Confidence Interval)**.** *Let* FAR *be the system's true (but unknown) false acceptance rate. For a fixed* $\gamma \in (0\%, 100\%]$ *let* $\mathrm{FAR}_0 \leq \mathrm{FAR}_1$ *such that* $\mathrm{FAR}^* \in [\mathrm{FAR}_0, \mathrm{FAR}_1]$ *for* $100\gamma\%$ *of all point estimations* FAR^**. The interval* $[\mathrm{FAR}_0, \mathrm{FAR}_1]$ *is called* γ-confidence interval *for* FAR*.* γ *is called* confidence level[8] *of the interval* $[\mathrm{FAR}_0, \mathrm{FAR}_1]$*.*

There are methods that compute confidence intervals for a given confidence level γ when s false accepts within N impostor recognition attempts have been observed. These are, for instance, the *Clopper-Pearson intervals* [40].

Example 4. *The* 95%*-Clopper-Pearson confidence interval for the false acceptance rate in Example 3 is* $[0.36\%, 0.81\%]$*, i.e. if* $s = 27$ *false accepts*

[8] A popular choice for a confidence level is $\gamma = 95\%$.

Table 4: Performance of the false-accept attack against the implementation of Section 2 using the parameters of the performance evaluation. The timings are have been determined on a 3.2Ghz desktop computer with four processor cores.

length of secret pol. k	point estimation of the false acceptance rate FAR*	avg. impostor decoding time IDT	95%-confidence interval for the false acceptance rate $[\mathrm{FAR}_0, \mathrm{FAR}_1]$	**expected time for a successful false-accept attack**	expected time for a successful brute-force attack
$= 7$	$188/4,856$	0.08 *sec*	$[3.33\%, 4.45\%]$	0.31 *sec*–0.41 *sec*	≈ 11.4 *sec*
$= 8$	$79/4,856$	0.140 *sec*	$[1.29\%, 2.02\%]$	1.19 *sec*–1.87 *sec*	≈ 2.97 *min*
$= 9$	$27/4,856$	0.198 *sec*	$[0.37\%, 0.81\%]$	4.22 *sec*–9.33 *sec*	≈ 49.4 *min*
$= 10$	$8/4,856$	0.240 *sec*	$[0.07\%, 0.32\%]$	12.8 *sec*–58.4 *sec*	≈ 13.9 *hours*
$= 11$	$5/4,856$	0.248 *sec*	$[0.03\%, 0.24\%]$	17.9 *sec*–2.14 *min*	≈ 10.2 *days*
$= 12$	$0/4,856$	0.193 *sec*	$[0.00\%, 0.06\%]$	> 54.2 *sec*	≈ 6.6 *months*

among $N = 4,856$ impostor recognition attempts have been observed. As a consequence, with a confidence of 95%, the expected time needed to perform a successful false-accept attack is between ≈ 4.23 sec and ≈ 9.33 sec.

3.3.2 Rule of Three

Assume we observed $s = 0$ false accepts among N impostor recognition attempts. Even if a point estimation yields a false acceptance rate of 0% this estimation is not very confident. The *rule of three* enables an easy way to estimate a 95%-confidence interval in this case (see [41, 42]).

Theorem 1 (Rule of Three)**.** *The interval $[0, 3/N]$ is a confidence interval of confidence level at least 95%.*

3.3.3 Evaluation

Example 5. *Assume an adversary has intercepted a minutiae fuzzy vault as in Section 2 with $k = 12$. The rule of three states that (with confidence 95%) we can only expect the true false-acceptance rate to be $\mathrm{FAR} \approx 0.06\%$. Thus, with an impostor decoding time of $\mathrm{IDT} \approx 0.193$ sec using Formula (5) the adversary can expect to successfully break the vault after ≈ 3 min 37 sec. If four processors/cores are used in parallel he may be successful even after ≈ 54.2 sec.*

For the implementation in Section 2 we estimated the expected computational time of a successful false-accept attack. The way we estimated the expected times is analogous to the estimations in Example 4 and Example 5. The results can be found in Table 4.

3.3.4 Evaluation against Alignment-Free Fuzzy Fingerprint Vault

For the alignment-free fuzzy fingerprint vault implementation of Li et al. (2010) [25] where $(n, t, k) = (440, 40, 13)$ a false-acceptance rate of 0.04% was reported.[9] The authors estimate the false-acceptance rate as a point estimation by observing $N = 34,650$ impostor authentication attempts.[10] Therefore, we assume that $s = 14$ false-accepts were observed in their experiment. The 95%-Clopper-Pearson confidence interval for the false-acceptance rate thus is $[0.0221\%, 0.0678\%]$. Furthermore, the authors report an average decoding time of 0.192 *sec*.[11] Consequently, using Formula (5) a false-accept attack may consume between 49 *sec* and 2.51 *min* of computer time if four processors/cores are used. In comparison to the brute-force, which is expected to require $\approx$ 20 *years* (see Table 2), the time for a successful false-accept attack is negligible. Moreover, the time is far away from being acceptable for a secure system.

For $k = 14$ no false-accepts were observed by the authors. The rule of three (see Section 3.3.2) states that (with a confidence of 95%) the true false acceptance rate is $< 0.0087\%$. Assuming IDT = 0.192 *sec* we can only expect the false-accept attack to require approximately 6.4 *min* which strongly contrasts an alleged security of 52 bits.

3.3.5 Evaluation against Fuzzy Vault with Minutiae Descriptors

If a vault was intercepted by an intruder in where the ordinate values are protected with minutiae descriptors (see Section 2.6) the false-accept attack can be run without modifications. For example, if $k = 12$ using the $(15, 5)$-BCH code, no false accepts have been observed within $9,900$ impostor authentication attempt in [23].[10] Thus, with the rule of three we can only expect the true false acceptance rate to be FAR $\leq 0.03\%$. By our experiments (see Table 1) we assume an average impostor decoding time of IDT = 0.193 *sec*. Consequently, we can only expect a false-accept to last $\log(0.5)/\log(1 - \text{FAR}) \cdot \text{IDT} \approx 7$ *min*. If all four processors are used in parallel, the time furthermore reduces to ≈ 2 *min*.

Let us discuss another interesting point. In [23] it is reported that if the $(511, 19)$-BCH code is used to protect the ordinate values of the construction

[9] We refer to Table 3 and 4 in [25] in where the *sum rule* is used for similarity measurement between vault features and query features.

[10] Actually, these are not statistically independent.

[11] The decoding times were reported for genuine authentication attempts only. For simplicity, we assume that it agrees with impostor decoding time.

of [20] the false acceptance rate drops from 0.7% to 0.01%. At a first glance, this may lead to the conclusion that the security is improved by a factor of ≈ 70. But this is not true: An adversary may decouple the basic vault construction from the protected ordinate values due to a very low sphere packing density. More precisely, the expected number of a protected vault point's ordinate value is estimated as $S = 1 + 4.25 \cdot 10^{-29}$ (see Section 3.2.2). Assuming that each vault point's correct ordinate value is contained in the candidate set, we set $S' = S - 1$ as the expected number of wrong ordinate values. Using *Markov's inequality*, the probability that there is at least one wrong candidate is less than $S' = 4.25 \cdot 10^{-29}$. Thus, the probability that decoupling the protected ordinate values to yield an instance of the basic vault without protected ordinate values is $(1 - S')^n = (1 - S')^{224} \approx 1 - 9.52 \cdot 10^{-27}$ which is overwhelming. Consequently, if the sphere packing density of the underlying error-correcting code is too small, protecting the ordinate values causes virtually no improvement against the false-accept attack.

3.4 Intermediate Discussion

Our investigations clearly show that biometric cryptosystems that are based on a single fingerprint cannot provide sufficient security — unless the false acceptance is reduced to a cryptographic negligible level: It is very easy to break a single fuzzy fingerprint vault using the false-accept attack. This highly advocates that a secure fingerprint cryptosystem must be based on multiple finger — or even finger in combination with other biometric modalities.

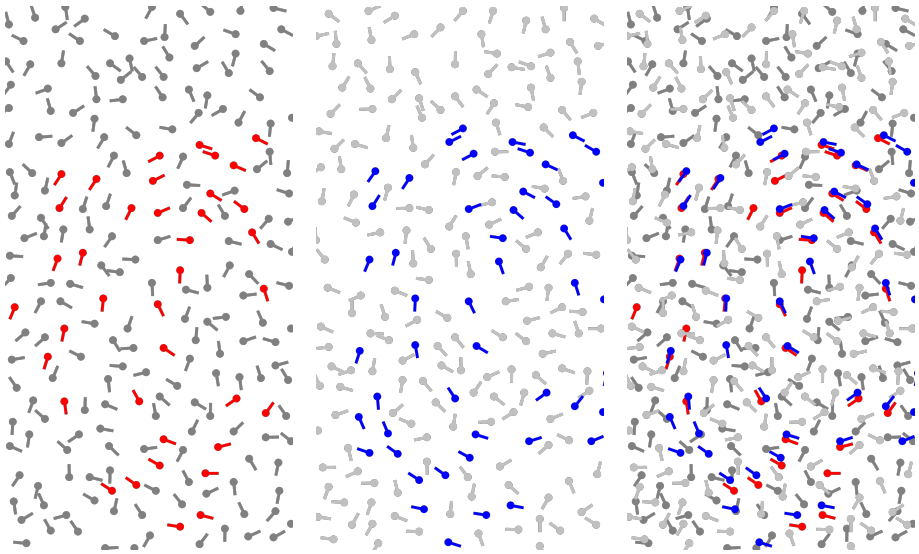
There remain problems with the fuzzy fingerprint vault that can not be solved merely by switching to multiple fingers and that have to be resolved first. These are the problems of cross-matching and the correlation attack.

3.5 Cross-Matching and the Correlation Attack

One of the most serious risks the fuzzy fingerprint vault is concerned with is its high vulnerability to cross-matching.

Cross-matching is always possible by, for instance, the brute-force attack: One of the vaults is attacked to reveal its template; this template is then used to open the other vault; if successful, both vaults are considered to match. While such an approach is always possible, there exists a more efficient method to separate genuine points from chaff points, if two vaults protecting the same finger are given: By correlating the vaults, genuine minutiae have a bias to be in agreement in both vaults while chaff minutiae are likely to be separate. An example illustrating this approach is given by Figure 3.

While correlation has the inadvertent effect that vault records can be



(a) (b) (c)

Figure 3: (a), (b) Two aligned vaults with chaff minutiae (gray and light-gray) and genuine minutiae (red and blue). (c) The genuine minutiae have a bias to be in agreement.

cross-matched, it is even possible for the attacker to efficiently break two vault's templates and keys given the vaults protect templates from the same finger as it was first supposed by Scheirer and Boult in 2007 [31]. As a consequence, separating genuine points from chaff points via correlation has the potential to be much more efficient than merely attacking one of the vaults via brute-force.

In 2008, Kholmatov and Yanikoglu [32] have demonstrated the effectiveness of the correlation attack. Against the fuzzy vault construction of Uludag et al. (2005) [18], they experimentally observed 59% successful recoveries using the correlation attack against 200 matching vault correspondences. Moreover, the authors were able to perform the correlation attack within 50 *sec* on average using a non-optimized Matlab implementation on a 3`Ghz` CPU. In comparison to the brute-force attack, which is expected to last $\approx$ 80 *hours* on a single core of a 3.2`Ghz` desktop computer (see Table 2), an intruder who has intercepted two matching vault records from different applications may quickly recover the corresponding templates and keys — even if he has no large database to perform a false-accept attack.

Cross-matching might already be enabled just by matching alignment helper data (see [19, 20, 34]) even though this alone does not imply that multiple records of the same template can be broken efficiently. While the possibility of cross-matching using alignment data alone is already an security issue, it is especially an issue in combination with the correlation

attack: An adversary may filter out genuine vault correspondences from different application's databases with the help of the public alignment data; afterwards, he can perform the correlation attack even faster because he can quickly align the vaults using the alignment data. Moreover, merely using alignment-free features as proposed by Li et al. (2010) [25] will not resolve the risk of cross-matching or attacks via record multiplicity.

Nandakumar, Nagar, and Jain (2008) [21] proposed to incorporate an additional user password into the vault. Furthermore, the additional security provided by the user passwords may prevent the vaults from being cross-matched and from being vulnerable to the correlation attack. However, using a user password causes inconveniences that were actually meant to be resolved by biometric based authentication schemes (e.g., weak or forgotten passwords).

In the next section we show that it is possible to implement a usable fingerprint fuzzy vault that is resistant against the correlation attack and that gets along without an additional user password.

4 Implementation of a Cross-Matching Resistant Minutiae Fuzzy Vault

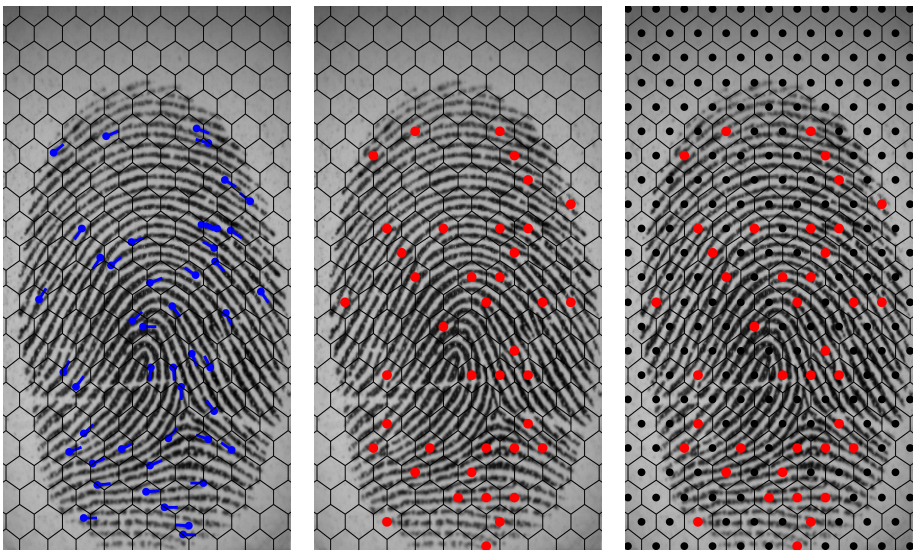
We have shown that a single finger is not sufficient to provide a secure biometric cryptosystem due to a cryptographically non-negligible false acceptance rate. Rather biometric cryptosystems that are based on multiple finger/modalities should be developed and analyzed more extensively. First steps have already been made (e.g., see [29,30]). However, it is obvious that merely fusing multiple finger to be protected by the fuzzy vault scheme will not resolve the problem of cross-matching or the correlation attack.

In this section, we propose an implementation of a minutiae fuzzy vault that is inherently resistant against cross-matching and that gets along without an additional password (see [21]). Roughly speaking, we achieve cross-matching resistance using the simple idea of rounding minutiae to a rigid hexagonal grid; the minutiae angles are quantized as well. Each element of the rigid system to where a minutia is quantized encodes a genuine vault point while the remaining elements encode chaff points. As a consequence the feature set between different vault records are equal which makes cross-matching via correlation useless to attack the vaults.

4.1 Vault Construction

Minutia Quantization

Given a minutiae template of a fingerprint, each of its minutia is quantized first. Let $\mathfrak{m} = (a, b, \theta)$ be a minutia at pixel (a, b) and of angle $\theta \in [0, 360)$. Let R_i be the point of a (hexagonal) grid $\{R_0, \ldots, R_{r-1}\}$ laying within the



(a) (b) (c)

Figure 4: The minutiae (a) positions are rounded to the location of the points of a hexagonal grid (b). Each other point of the grid (c) is used to encode a chaff point.

fingerprint image's region that best approximates (a, b). Furthermore, let $j = \lfloor \theta/360 \cdot s \rfloor$ where s denotes the parameter controlling the number of values into where angles are quantized. Now, the integer $i + r \cdot j$ encodes the quantization of $\mathfrak{m}$. Let $x_{i,j} \in \mathbf{F}$ denote the finite field element encoding $i+r\cdot j$ by some (but fixed) convention. Then the quantization of the minutia $\mathfrak{m}$ is given by the map $\mathrm{quant}(\mathfrak{m}) \mapsto x_{i,j}$.

Note, that the feature set in where minutiae quantizations can occur is $\mathbf{E} = \{\ x_{i,j} \mid i = 0, \ldots, r-1,\ j = 0, \ldots, s-1\}$.

Enrollment

Let $t_{\max}$ be a bound on the genuine point's size and T be an input minutiae template that we want to protect. Write $T = \{\mathfrak{m}_1, \mathfrak{m}_2, \ldots\}$ and assume that if $\mathfrak{m}_i$ is of better quality than $\mathfrak{m}_j$ this implies $i < j$. The feature set $\mathbf{A}$ is defined to contain at most $t_{\max}$ quantizations of the first best-quality minutiae. Note, that $\mathbf{A}$ can contain fewer elements than T.[12] Let $t = |\mathbf{A}|$.

The next step is to bind the template quantized as $\mathbf{A}$ to a secret polynomial $f \in \mathbf{F}[X]$ of degree $< k$. This is done as usual by letting the genuine set $\mathbf{G} = \{\ (x, f(x)) \mid x \in \mathbf{A}\ \}$.

[12] If there are minutiae in T that have equal quantization then it is possible that $|\mathbf{A}| < |T|$.

As in the usual vault, the genuine set is hidden among a large set of chaff points. But now, every element in $\mathbf{E}$ that is not contained in $\mathbf{A}$ corresponds to a chaff point. More precisely, $\mathbf{C} = \{ (x, y) \mid x \in \mathbf{E} \setminus \mathbf{A} \}$ where the ys are chosen uniformly at random from $\mathbf{F}$ such that $y \neq f(x)$.

The vault consists of the union of genuine and chaff points. Furthermore, a cryptographic hash value of the secret polynomial is stored along with the vault. Thus the public vault is the tuple $(\mathbf{V}, h(f))$ where $\mathbf{V} = \mathbf{G} \cup \mathbf{C}$.

Vault Authentication

On authentication, a query minutiae template is given for which we assume that it is already aligned to the vault. Then the corresponding feature set $\mathbf{B}$ is extracted from the query template in the same way as $\mathbf{A}$ was extracted from the enrollment template. Using $\mathbf{B}$ the unlocking set is built out of those points from $\mathbf{V}$ that have abscissa value in $\mathbf{B}$, i.e. $\mathbf{U} = \{ (x, y) \in \mathbf{V} \mid x \in \mathbf{B} \}$.

Let $\omega = |\mathbf{A} \cap \mathbf{B}|$. Then $\mathbf{U}$ contains exactly ω genuine points. Thus, if $\omega \geq k$ the secret polynomial f can be obtained from $\mathbf{U}$ in the same way as described in Section 2.2.

4.2 Training

Our construction is controlled by the following parameters:

- The minimal distance of the hexagonal grid points λ which (together with the fingerprint image's dimension) controls the number of hexagonal grid points r;
- the number of values s into where the minutia's angle are quantized;
- the bound $t_{\max}$ on the number of genuine vault points;
- the size k of the secret polynomial.

We performed systematical tests to determine a good configuration of the above parameters. Therefore we determined the GARs and the FARs on the FVC 2002 DB2-B (which is intended for training purposes; see [35]) for each configuration of

$$\begin{aligned} \lambda = 8, \ldots, 32; \quad s = 1, \ldots, 8; \\ t_{\max} = 10, \ldots, 60; \quad k = 1, \ldots, t_{\max}. \end{aligned} \tag{6}$$

For the GARs, each finger's ith impression was used to extract the feature set $\mathbf{A}$; each jth (where $j > i$) impression of the finger aligned to the ith was used to extract its features $\mathbf{B}$; if $|\mathbf{A} \cap \mathbf{B}| \geq k$ this was accounted for as a genuine accept; otherwise it was accounted for as a false reject. For the FAR, each Ith finger's first impression was used to extract the feature set $\mathbf{A}$; each Jth finger's first impression (where $J > I$) was used to extract

$\mathbf{B}$; again, if $|\mathbf{A} \cap \mathbf{B}| \geq k$ this was counted as a false accept; otherwise as a reject.

The best configuration[13] was obtained as

$$\lambda = 29,\ s = 6,\ t_{\max} = 44,\ \text{and}\ k = 7 \tag{7}$$

with a GAR = 100% and FAR = 0%. The number of hexagonal grid points of minimal distance $\lambda = 29$ that fit in an image of dimension 296×560 is $r = 242$. For the angles are quantized into $s = 6$ possible values, the size of the vault is $n = r \cdot s = 1452$. Thus, the brute-force security is $\mathbf{bf}(1452, 44, 7) \approx 2^{36}$. If a higher security is sought, we may choose a higher k.

4.3 Randomized Decoder

The potential recognition performance of our construction looks promising. However, there remains a problem concerning the decoding work. If a brute-force security at least 2^{40} is sought we may choose $k = 8$. On an authentication attempt, an unlocking set of size up to $t_{\max} = 44$ is built. If we would attempt to decode by iterating through all candidate polynomials of degree $< k$ that interpolate k unlocking points, $\binom{44}{8} \approx 2^{27}$ iterations have to be performed in the worst case before the user is possibly accepted/rejected. This is too expensive for a usable system.

As a countermeasure, we propose to randomize the decoding procedure of Section 2.2. Instead of iterating through all polynomials of degree $< k$ that interpolate k unlocking points, we only iterate through at most $\mathcal{D}$ polynomials each interpolating k randomly selected unlocking points.

On authentication, if the unlocking set $\mathbf{U}$ contains $\omega \geq k$ genuine points the randomized decoder will successfully output the correct polynomial with probability at least $1 - (1 - \mathbf{bf}(t_{\max}, \omega, k)^{-1})^{\mathcal{D}}$ which approaches 100% as $\mathcal{D} \to \infty$. Moreover, if the unlocking set $\mathbf{U}$ contains $\omega < k$ genuine points the randomized decoder will not succeed in decoding. Consequently, both GAR and FAR for a fixed k drop if the randomized decoder is used.

Furthermore, the use of the randomized decoder only affects authentication and not the vault construction. Thus, for a fixed k, the overall security does not suffer if the randomized decoder is used.

4.4 Performance Evaluation

For the configuration determined during the training and for different k, we performed performance evaluations of our implementation following the description of Section 2.4. We have chosen $\mathcal{D} = 2^{16}$ iterations for the randomized decoder which corresponds to a reasonable amount of iterations

[13] The best configuration was defined as to yield the highest GAR at the lowest FAR; among these configurations, the one with maximal $k/t_{\max}$ has been selected.

Table 5: Result of the Performance Evaluation

pol. degree	genuine acceptance rate	false acceptance rate	avg. genuine decoding time	avg. impostor decoding time	brute-force security
$< k$	GAR (sub-GAR)	FAR	GDT	IDT	$\mathbf{bf}(1452, 44, k)$
$= 7$	$\approx 91.96\%$ ($= 97\%$)	$\approx 0.46\%$	≈ 0.03 *sec*	≈ 0.28 *sec*	$\approx 2^{36}$
$= 8$	$\approx 86.82\%$ ($= 95\%$)	$\approx 0.081\%$	≈ 0.06 *sec*	≈ 0.35 *sec*	$\approx 2^{43}$
$= 9$	$\approx 80.36\%$ ($= 93\%$)	$\approx 0.02\%$	≈ 0.1 *sec*	≈ 0.41 *sec*	$\approx 2^{47}$
$= 10$	$\approx 72.61\%$ ($= 91\%$)	$= 0\%$	≈ 0.17 *sec*	≈ 0.51 *sec*	$\approx 2^{52}$
$= 11$	$\approx 63.11\%$ ($= 86\%$)	$= 0\%$	≈ 0.26 *sec*	≈ 0.60 *sec*	$\approx 2^{57}$
$= 12$	$\approx 53.57\%$ ($= 80\%$)	$= 0\%$	≈ 0.39 *sec*	≈ 0.73 *sec*	$\approx 2^{63}$

that is feasible on current hardware. In addition to genuine acceptance rates and false acceptance rates, the average genuine decoding times as well as the average impostor decoding times were determined. The results can be found in Table 5.

In order to compare our results with other fuzzy vault implementations, we also kept track of the genuine acceptance rate in which only the first two impressions are taken into account (the first impression is used for enrollment and the second as the query). The corresponding rates are denoted as sub-GAR in Table 5. We reached sub-GAR = 91% in the case no false accepts have been observed. In comparison, on the same dataset Nagar et al. (2010) [23] achieve sub-GAR = 93% at zero false accepts while Li et al. (2010) [25] achieve sub-GAR = 92%. Even though our results are only valid under a well-solved alignment framework our implementation provides resistance against the correlation attack — even without a user password (see [21]).

4.5 Alternative Fuzzy Extractor Construction

Another advantage of our implementation is that it can be easily modified to meet the requirements for the modified fuzzy vault construction proposed by Dodis et al. (2008) [14]. This construction avoids the generation of chaff points and significantly reduces the amount of memory that is required for storage. The changes that have to be made would not affect the construction's performance or security against the brute-force or false-accept attack. For details of the construction we refer to [14].

However, without preventions, multiple records of the fuzzy extractor construction may become vulnerable to cross-matching, especially, if the

protected templates are equal. The way of cross-matching is similar to the *decodability attack* as it has been investigated by Kelkboom et al. (2011) [43]. Fortunately, applying a random bit-permutation process secures the fuzzy extractor construction from cross-matching based on the decodability attack. For details we refer to [43].

4.6 Security Analysis

Our implementation provides good security against the brute-force attack. For example, if $k = 10$ at a 52-bit brute-force security level we have empirically determined that an adversary can test $128,205$ polynomials per second on a single core of a 3.2`Ghz` desktop computer with four processor cores. Thus, if all four cores are used in parallel, he can expect to break an instance of our implementation after approximately 192 *years*.

Our implementation obviously is resistant against the correlation attack and cross-matching via correlation. But the implementation's vulnerability against the false-accept attack remains to be evaluated.

It is possible to analyze the false-accept attack analogous to Section 3.3 using confidence intervals. But there is a more elegant way to estimate the false-acceptance rate.

Assume that within an impostor authentication attempt an unlocking set of size t is built containing ω genuine vault points. Using $\mathcal{D}$ decoding iterations the vault can be unlocked with probability

$$p(t, \omega, \mathcal{D}) = \begin{cases} 1 - (1 - \mathbf{bf}(t, \omega, k)^{-1})^{\mathcal{D}}, & \text{if } \omega \geq k \\ 0, & \text{if } \omega < k. \end{cases}$$

Thus, if in a test with N impostor authentication attempt the ith unlocking set is of size t_i containing ω_i genuine vault points then we may estimate the false acceptance rate as $\text{FAR} \approx \frac{1}{N} \sum_{i=1}^{N} p(t_i, \omega_i, \mathcal{D})$. Note, that the effort in authenticating increases linearly with the number of decoding iterations $\mathcal{D}$. Therefore, we estimate the cost for a successful false-accept attack as $\log(0.5)/\log(1 - \text{FAR}) \cdot \mathcal{D}$.

As the attacker is free in choosing whichever decoder he prefers, he may choose the number of decoding iterations that minimizes the cost.

Lemma 1. *The cost for a successful false-accept attack is minimized for* $\mathcal{D} = 1$.

Proof. Let $\epsilon(x)$ be the false-acceptance rate as a function in the number of decoding iterations x. Then $g(x) = \log(0.5)/\log(1 - \epsilon(x)) \cdot x$ is the cost function of a successful false-accept attack. Note that we can write $1 - \epsilon(x) = \frac{1}{N} \sum \alpha_i^x$ where $0 \leq \alpha_i \leq 1$. Using *Jensen's inequality* we can

bound $1-\epsilon(x) \geq \left(\frac{1}{N}\sum \alpha_i\right)^x$. Thus,

$$\begin{aligned} g(x) &= \frac{|\log(0.5)|}{|\log(1-\epsilon(x))|} \cdot x \geq \frac{|\log(0.5)|}{\left|\log\left(\left(\frac{1}{N}\sum \alpha_i\right)^x\right)\right|} \cdot x \\ &= \frac{|\log(0.5)|}{\left|\log\left(\frac{1}{N}\sum \alpha_i\right)\right| \cdot x} \cdot x = \frac{|\log(0.5)|}{\left|\log\left(\frac{1}{N}\sum \alpha_i\right)\right|} = g(1) \end{aligned}$$

which proves the lemma. □

The lemma enables us to estimate a lower bound for the cost of the false-accept attack against our implementation assuming the adversary also utilizes the randomized decoder. However, the attacker may prefer to use more than only one decoding iteration, e.g., if he uses a fingerprint database for the attack of medium size. Furthermore, the time needed to build the unlocking sets was not taken into account, but it increases the cost for a false-accept attack in practice. But for a security analysis, it is safer to rely on a lower bound.

Hence, in a test of N impostor authentication attempts in where the ith unlocking set was of size t_i containing ω_i genuine points, the cost for a successful false-accept attack can be estimated as $\log(0.5)/\log(1-\text{FAR})$ where $\text{FAR} = \frac{1}{N}\sum p(t_i, \omega_i)$ with

$$p(t_i, \omega_i) = p(t_i, \omega_i, 1) = \begin{cases} \mathbf{bf}(t_i, \omega_i, k)^{-1} & \text{if } \omega_i \geq k \\ 0 & \text{if } \omega_i < k. \end{cases}$$

Table 6: Performance of the false-accept attack on a four-core desktop computer with 3.2Ghz.

polynomial degree $< k$	false acceptance rate FAR	**expected time for a false-accept attack**
$= 7$	$\approx 8.31 \cdot 10^{-8}$	≈ 36 *sec*
$= 8$	$\approx 8.87 \cdot 10^{-9}$	≈ 2 *min*
$= 9$	$\approx 8.53 \cdot 10^{-10}$	≈ 21 *min*
$= 10$	$\approx 6.95 \cdot 10^{-11}$	≈ 5 *hours*
$= 11$	$\approx 4.40 \cdot 10^{-12}$	≈ 4 *days*
$= 12$	$\approx 1.86 \cdot 10^{-13}$	≈ 120 *days*

To estimate the false acceptance rate of our implementation for different k, we simulated impostor authentication attempts on the FVC 2002 DB2-A

database following the FVC protocol which yielded $N = 4,950$ impostor authentication attempts. For the simulations, we used the same configurations as in the performance evaluation. For the ith impostor authentication attempt, we quantized the first finger as the set $\mathbf{A}$ and the second as $\mathbf{B}$. Then we let $t_i = |\mathbf{B}|$ and counted $\omega_i = |\mathbf{A} \cap \mathbf{B}|$. Using $p(t_i, \omega_i)$ we estimated the false acceptance rate FAR for a single decoding iteration and the corresponding cost for the false-accept attack. Consulting the impostor decoding times determined during the evaluation we know how much 2^{16} iterations of the false-accept attack cost. We used this information to determine the time for a successful false-accept attack on a 3.2`Ghz` desktop computer. The results can be found in Table 6.

Example 6. *For example, if $k = 9$ the false acceptance rate was found to be* $\mathrm{FAR} \approx 8.53 \cdot 10^{-10}$ *for a single decoding iteration. Thus, the attacker can expect to use approximately* $\log(0.5)/\log(1 - 8.53 \cdot 10^{-10}) \approx 8.13 \cdot 10^8$ *finger as queries to successfully break the vault. As the time for* 2^{16} *iterations was found to be* $\mathrm{IDT} \approx 0.41$ *sec the time for a successful false-accept attack can be estimated as* $8.13 \cdot 10^8 / 2^{16} \cdot 0.41$ *sec* ≈ 1–2 *hours. If all four processor cores were used in parallel, the time furthermore reduces to* ≈ 21 *min which is much more efficient than the brute-force attack requiring* 192 *years. Please note, like other implementations our construction is also vulnerable to intensive false-accept attacks.*

5 Discussion and Outlook

We investigated the security of current implementations of the fuzzy fingerprint vault. We found that, even if the brute-force attack is impractical against some implementations, this does not hold for the false-accept attack. This attack is feasible for every authentication scheme in which the false acceptance rate is non-negligible and thus it is for current implementations of biometric cryptosystem protecting a single fingerprint's template. Even worse, according to our observations, the false-accept attack can be performed much more efficiently than the brute-force attack. One may argue, that it is infeasible for an adversary to establish databases which are of sufficient size to perform intensive false-accept attacks off-line. First, in our view, this can not be prevented having in mind that there exist large databases containing real fingers. Second, the performances of false-accept attacks are hints for the existence of similar efficient statistical attacks. Such attacks may be prevented using multiple fingers or even multiple biometric modalities. Therefore, multi-finger fuzzy vaults should be investigated as a potential method wherever high security is important. And yet a significant risk remains: The correlation attack cannot be prevented merely by using multiple fingers.

Therefore we endeavored to solve the problem of the correlation attack.

In this paper we have demonstrated that it is possible to implement a minutiae fuzzy vault that is resistant against the correlation attack without loss of authentication performance. Our implementation primarily relies on the simple innovation of rounding minutiae to a rigid grid while using the entire grid as vault features, thereby preventing attackers from distinguishing genuine from chaff features via correlation. Furthermore, to make vault authentication practical, we proposed to use a randomized decoder rather than systematically iterating through all candidate polynomials. Since the randomized decoder only affects vault authentication and not vault construction, the randomized decoder does not adversely affect vault security. Well conceived, the randomized decoder may be incorporated into a wide variety of fuzzy vault implementations, not only fuzzy vaults with the express purpose of protecting minutiae templates of a single finger. Furthermore, our single-finger fuzzy vault construction that is resistant against the correlation attack may be generalized to a construction that protects multiple fingers.

All experiments described in this paper can fully be reproduced using software available for download.[14]

We did not propose a mechanism for dealing with alignment for our vault construction. Although it would have been possible to adopt the ideas available in the literature that propose to store additional alignment-helper data publicly with the vault [17, 19, 20, 33, 34] it is not yet clear how this would affect vault security. Moreover, some of the proposals find accurate alignment via multiple candidate alignments: During authentication, for each candidate alignment an authorization attempt is performed until the correct secret is seen. Translating this method to multiple fingers is problematic because the number of candidate alignments grows exponentially with the number of fingers. Consequently, fingerprint alignment techniques for multi-finger fuzzy vaults should be reconsidered.

Ideally, fingerprints could be pre-aligned. This would make iterations through several candidate alignments obsolete. Moreover, fuzzy vaults protecting accurately pre-aligned fingers do not need to store additional alignment-helper data which can cause unwanted information leakage regarding the corresponding finger. Prealignment of fingerprints is strongly related to the concept of intrinsic coordinate systems [44, 45]. Unfortunately, current methods that extract intrinsic coordinate systems are not robust enough to produce fingerprint pre-alignment of sufficient accuracy. Although challenging, it may be worthwhile to seek more robust methods to extract intrinsic

[14] This comprises performance analyses of the brute-force attack, performance evaluations of the minutiae fuzzy vault implementation as in Section 2, the training for determining a good configuration of our cross-matching resistant minutiae fuzzy vault implementation, its performance evaluations, and a program to analyze our implementation's resistance against the false-accept attack; these are sample programs for a C++ software library that we call `thimble`; visit `http://www.stochastik.math.uni-goettingen.de/biometrics` for downloading its source code.

coordinate systems.

References

[1] A. Juels and Sudan, "A fuzzy vault scheme," in *Proc. IEEE Int'l Symp. Inf. Theory*, A. Lapidoth and E. Teletar, Eds., 2002, p. 408.

[2] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *CCS '99: Proc. of the 6th ACM Conf. on Computer and Communications Security*. New York, NY, USA: ACM, 1999, pp. 28–36.

[3] D. Gorenstein, W. W. Peterson, and N. Zierler, "Two-error correcting bose-chaudhuri codes are quasi-perfect," *Information and Control*, vol. 3, no. 3, pp. 291–294, 1960.

[4] E. Berlekamp, *Nonbinary BCH decoding*, ser. Institute of Statistics mimeo series. University of North Carolina. Dept. of Statistics, 1966.

[5] J. L. Massey, "Shift-register synthesis and bch decoding." *IEEE Trans. Inf. Theory*, vol. 15, no. 1, pp. 122–127, 1969.

[6] S. Gao, "A new algorithm for decoding reed-solomon codes," in *in Communications, Information and Network Security, V.Bhargava, H.V.Poor, V.Tarokh, and S.Yoon*. Kluwer, 2002, pp. 55–68.

[7] M. Sudan, "Decoding of reed solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, pp. 180–193, 1997.

[8] V. Guruswami and M. Sudan, "Improved decoding of reed-solomon and algebraic-geometric codes," *IEEE Trans. Intell. Transp. Syst.*, vol. 45, pp. 1757–1767, 1998.

[9] D. Bleichenbacher and P. Q. Nguyen, "Noisy polynomial interpolation and noisy chinese remaindering," in *Proc. Int. Conf. on Theory and application of cryptographic techniques*, ser. EUROCRYPT'00, Berlin, Heidelberg, 2000, pp. 53–69.

[10] V. Guruswami and A. Vardy, "Maximum-likelihood decoding of reed-solomon codes is np-hard," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '05. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005, pp. 470–478.

[11] A. Kiayias and M. Yung, "Cryptographic hardness based on the decoding of reed-solomon codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2752–2769, Jun. 2008.

[12] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of Biometrics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[13] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, 2nd ed. Springer Publishing Company, Incorporated, 2009.

[14] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, 2008.

[15] A. Arakala, J. Jeffers, and K. J. Horadam, "Fuzzy extractors for minutiae-based fingerprint authentication," in *ICB*, ser. Lecture Notes in Computer Science, S.-W. Lee and S. Z. Li, Eds., vol. 4642. Springer, 2007, pp. 760–769.

[16] T. C. Clancy, N. Kiyavash, and D. J. Lin, "Secure smartcardbased fingerprint authentication," in *Proc. ACM SIGMM workshop on Biometrics methods and applications*, ser. WBMA '03. New York, NY, USA: ACM, 2003, pp. 45–52.

[17] S. Yang and I. Verbaudwhede, "Automatic secure fingerprint verification system based an fuzzy vault scheme," in *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, 2005, pp. 609–612.

[18] U. Uludag, S. Pankanti, and A. Jain, "Fuzzy vault for fingerprints," in *Proc. Int'l Conf. on Audio- and Video-Based Biometric Person Authentication*, 2005, p. 310319.

[19] U. Uludag and A. Jain, "Securing fingerprint template: fuzzy vault with helper data," in *Proc. CVPR Workshop on Privacy Research In Vision*, 2006, p. 163.

[20] K. Nandakumar, A. K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 4, pp. 744–757, 2007.

[21] K. Nandakumar, A. Nagar, and A. K. Jain, "Hardening fingerprint fuzzy vault using password." in *ICB'07*, 2007, pp. 927–937.

[22] A. Nagar, K. Nandakumar, and A. K. Jain, "Securing fingerprint template: Fuzzy vault with minutiae descriptors," in *Proc. Int'l Conf. on Pattern Recognition (ICPR), Dec., 2008*, 2008.

[23] ——, "A hybrid biometric cryptosystem for securing fingerprint minutiae templates," *Pattern Recogn. Lett.*, vol. 31, pp. 733–741, June 2010.

[24] J. Feng, "Combining minutiae descriptors for fingerprint matching," *Pattern Recognition*, vol. 41, no. 1, pp. 342–352, 2008.

[25] P. Li, X. Yang, K. Cao, X. Tao, R. Wang, and J. Tian, "An alignment-free fingerprint cryptosystem based on fuzzy vault scheme," *J. Netw. Comput. Appl.*, vol. 33, pp. 207–220, May 2010.

[26] X. Jiang and W.-y. Yau, "Fingerprint minutiae matching based on the local and global structures," *Proc. Int. Conf. on Pattern Recognition ICPR*, vol. 2, pp. 1038–1041, 2000.

[27] P. Mihăilescu, A. Munk, and B. Tams, "The fuzzy vault for fingerprints is vulnerable to brute force attack," in *BIOSIG*, 2009, pp. 43–54.

[28] N. Boulgouris, K. Plataniotis, and E. Micheli-Tzanakou, *Biometrics: theory, methods, and applications*, ser. IEEE ser. on computational intelligence. IEEE Press, 2009.

[29] J. Merkle, H. Ihmor, U. Korte, M. Niesing, and M. Schwaiger, "Performance of the fuzzy vault for multiple fingerprints (extended version)," *CoRR*, vol. abs/1008.0807, 2010.

[30] A. Nagar, K. Nandakumar, and A. K. Jain, "Multibiometric cryptosystems based on feature-level fusion," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 255–268, 2012.

[31] W. J. Scheirer and T. E. Boult, "Cracking fuzzy vaults and biometric encryption," in *Proc. of Biometrics Symp.*, 2007, pp. 1–6.

[32] A. Kholmatov and B. Yanikoglu, "Realization of correlation attack against the fuzzy vault scheme," in *Proc. SPIE*, 2008.

[33] J. Jeffers and A. Arakala, "Fingerprint Alignment for a Minutiae-Based Fuzzy Vault," in *Proc. Biometrics Symp.*, 2007, pp. 1–6.

[34] J. Li, X. Yang, J. Tian, P. Shi, and P. Li, "Topological structure-based alignment for fingerprint Fuzzy Vault," in *Proc. Int'l Conf. on Pattern Recognition*, 2008, pp. 1–4.

[35] D. Maio *et al.*, "FVC2002: Second Fingerprint Verification Competition," in *Proc. Int'l Conf. on Pattern Recognition (ICPR2002), Quebec City*, 2002, pp. 811–814.

[36] Neurotechnology Ltd, "Verifinger SDK 5.0," http://www.neurotechnology.com, 2006.

[37] C. Gottschlich, P. Mihailescu, and A. Munk, "Robust orientation field estimation and extrapolation using semilocal line sensors," *IEEE Trans. Inf. Forensics Security*, vol. 4, pp. 802–811, December 2009.

[38] C. Gottschlich, "Curved regions based ridge frequency estimation and curved gabor filters for fingerprint image enhancement." *IEEE Trans. Image Process.*, vol. 21, pp. 2220–2227, 2012.

[39] W. yong Choi, S. B. Pan, J.-M. Kim, Y. Chung, and D.-W. Hong, "Fast polynomial reconstruction attack against fuzzy fingerprint vault," in *Information Science and Service Science (NISS), 2011 5th International Conference on New Trends in*, vol. 2, 2011, pp. 299–302.

[40] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 1934.

[41] J. Hanley and A. Lippman-Hand, "If nothing goes wrong, is everything allright? interpreting zero numerators," *Journal of the American Medical Association*, vol. 249, no. 13, pp. 1743–1745, 1983.

[42] B. D. Jovanovic and P. S. Levy, "A look at the rule of three," *The American Statistician*, vol. 51, no. 2, pp. 137–139, 1997.

[43] E. J. Kelkboom, J. Breebaart, T. A. Kevenaar, I. Buhan, and R. N. Veldhuis, "Preventing the decodability attack based cross-matching in a fuzzy commitment scheme," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 107–121, March 2011.

[44] A. M. Bazen and S. H. Gerez, "An intrinsic coordinate system for fingerprint matching," in *Audio- and Video-Based Biometric Person Authentication*, ser. Lecture Notes in Computer Science, J. Bigun and F. Smeraldi, Eds., vol. 2091. London: Springer Verlag, June 2001, pp. 198–204.

[45] T. Hotz, "Intrinsic coordinates for fingerprints based on their longitudinal axis," in *Proc. Int'l Symp. on Image and Signal Processing and Analysis*, pp. 501–504.